Devin SWE-2 across every effort level

VulcanBench Frontier v4 | Code quality protocol v3.11 | September 2026

Abstract

Devin SWE-2 ran the 23-task VulcanBench Frontier v4 suite through the Devin CLI on a Devin subscription on September 18 to 21, 2026, once per task at each of the three effort levels the model offers (medium, high and max, its whole catalog), 69 runs in all. Code quality carries 33% of the combined score. It is judged here for a named human reader by **Muse Spark 1.3 (Meta) alone**, not by the two-judge neutral panel behind every other Frontier v4 entry: Grok 4.6 failed the calibration exam under protocol v3.9 and GPT-5.6 Sol, admitted under v3.10 to fill the second seat, failed under v3.10, both on gate 16, so the protocol's pre-registered single-panel rule applies. Read every Code quality number in this report as one judge's rating. Medium and high are indistinguishable: 82.43 and 81.42, with standard errors of 2.00 and 2.98, and both pass 15 of 23 tasks. Only max moves, to 86.14 and 21 of 23. Code quality runs 65.14 to 69.17 across the three levels. Cost per task is unavailable: Cognition publishes no per-token rate for SWE-2, so nothing here is priced. What the sweep spends is time and tokens, 35.1 to 56.6 minutes and 7.53M to 16.80M raw tokens per task.

65 of the 69 runs are judged. One high run (cellarcore) reached the 3-hour task budget before verification and did not finish; three more (snapcore and vaultcore at high, freightcore at max) changed no recognized source file, so there was no submission to review and the sweep's automated quality and security metrics are undefined for them. The protocol excludes those four rather than judging them. All four failed their tests, so the sweep's pass counts are unchanged by the exclusions. Under the prior 50/15/15/20 profile applied to the same Code quality scores the shape is the same, so the picture does not depend on the weight change.

Table 1. Combined score, Code quality, tasks passed and runtime by effort

Model	Effort	Combined (33%)	SE	Combined (20%)	Code quality	Min/task	Passed
SWE-2	Medium	82.43	2.00	85.46	65.14	49.8	15/23
SWE-2	High	81.42	2.98	84.01	66.26	35.1	15/20
SWE-2	Max	86.14	1.31	88.10	69.17	56.6	21/22

Combined (33%) = 0.50 functional + 0.085 lint and complexity + 0.085 security + 0.33 Code quality. Combined (20%) applies the prior 50/15/15/20 profile to the same Code quality scores for comparison. SE is one sample standard error across the cell's tasks, not judge uncertainty or a significance test. Passed counts judged tasks with a perfect functional score, out of the judged runs in the cell; because every excluded run failed, the sweep's own counts are 15, 15 and 21 of 23. Runtime is solver wall-clock per task over every finished run in the cell; judging is excluded.

The Code quality protocol

Two failure modes motivated the reviewed score. The lint and complexity metric rewards compression: the maintainability index carries a lines-of-code term and per-function complexity stays low when each dense line does something different, so five statements on a line with names like x and k can outscore the same logic written for a person. And a large model reading compressed code pays almost nothing to parse it; when a frontier model was calibrated as a readability judge, it rated a squashed six-line function the same as its formatted copy. Since September 7, 2026 Code quality carries 33% of the combined score and lint and complexity and security carry 8.5% each. The prior profile is reported beside the new one on every table.

Protocol code-quality-maintenance-v3.11 is the same protocol applied to this population. Nothing in the rubric, controls, quirk keys, gates, repeats, seed or judge settings changed. What did change is the population and the panel. The population is the sweep minus the four runs that carry no judgeable submission, frozen on September 21 with 65 rows, none missing. The panel is one judge, for the reason set out on the next page. Every card and report that shows Devin's Code quality is required by the amendment to say so.

The rubric

The judge receives instructions frozen by hash before any review. The prompt names the reader it scores for: an engineer who has never seen the code, reads it top to bottom without running it, and must make a correct change in one sitting. It tells the judge that its own ease at parsing dense code is not evidence of readability. Six dimensions are scored 0 to 4 in half steps against written anchors, each with an exact excerpt and a concrete consequence for that reader.

Sub-score	Dimension	What is assessed
Human readability	Naming	Identifiers say what things are in the task's domain
Human readability	Presentation	Statements per line, nesting, function length, how much the reader must hold at once
Human readability	Intent	Constants, thresholds and quirks explained by names or accurate comments that say why
Maintainability	Structure	Coherent responsibilities, no duplicated policy, no abstraction beyond the task's scale
Maintainability	Changeability	A named plausible change, every edit site traced, scored on locality and safety
Maintainability	Verifiability	Explicit state, failures that name what went wrong, seams to test one rule alone

The layers under the 33 points

Reviewed panel, 24 points. The six-dimension rubric above, from the single scored judge. **Intent recovery, 9 points.** Every task in this suite is a rewrite of a retired binary whose real behaviour departs from its written specification in documented ways. The judge sees only the specification and the code, never the issue text, and lists where the code departs from the specification; a separate call matches that list against the frozen answer key. The denominator is the quirks the submission actually passed tests for, so a functional failure is not punished twice; one high run passed none, so its Code quality is the reviewed score alone under the pre-registered redistribution. **Measured maintenance, designed for 12 points, not yet built.** Until it exists the pre-registered split above is in force and is stated on the card.

One judge, and why

The scored panel is Muse Spark 1.3 through the Muse CLI on its Standard tier, which does not train on prompts, and nothing else. Meta has no model on this board, so the judge does not grade a relative. Every session is fresh, tools are disabled, the workspace is empty, model identity is checked per call, and solver labels are withheld. This is the only Frontier v4 entry scored by a single judge; the rest carry the mean of Muse Spark 1.3 and Grok 4.6.

The calibration exam

Before scoring a single submission a judge reviews ten held-out programs that implement the same ledger specification: clear, compressed, auto-formatted, verbose, over-abstracted, misleadingly commented, over-narrated, quirky, prompt-injecting and stateful. Each is reviewed five times in a seeded order, and twenty gates fixed in advance check that the judge sees the construct; a judge may miss at most one gate by at most half a point.

Gate 16 is the probe on the clear control, and it is boolean: a program with no documented departure from its specification must draw an empty probe on at least four of five repeats. Grok 4.6 reported invented departures on two repeats under v3.9 and GPT-5.6 Sol on four under v3.10. Each passed every other gate, repeatability included, and because the gate is boolean the one-gate allowance cannot cover it. Under the pre-registered single-panel rule nothing further runs for a judge that fails, and no judge retakes a gate it failed. Both verdicts stay published. Muse's own verdict comes from the v3.9 freeze rather than a fresh exam: the exam is per judge and control set and neither changed between v3.9 and v3.11, so on the v3.6.1 precedent the runner reuses it.

Table 2. Calibration verdicts

Judge	Protocol	Calls	Result	Allowance used	Failing gates
Muse Spark 1.3, scored	code-quality-maintenance-v3.9	80	passed	yes	g11_repeatability
Grok 4.6, not scored	code-quality-maintenance-v3.9	80	failed	no	g16_probe_recovers_docum
GPT-5.6 Sol, not scored	code-quality-maintenance-v3.10	80	failed	no	g16_probe_recovers_docum

Muse passed 19 of the 20 gates outright and used the pre-registered one-gate allowance on the repeatability gate, within the allowance as written. Both failures are on gate 16, the one call that rewards saying nothing.

Table 3. Control means on three of the ten calibration programs

Judge	Control	Naming	Present.	Intent	Struct.	Change.	Verif.
Muse	clear	4.00	3.90	3.40	4.00	4.00	3.90
Muse	compressed	1.20	1.70	1.00	3.70	2.50	3.00
Muse	misleading comments	3.80	3.80	0.20	3.90	2.00	3.40
Grok	clear	3.70	3.60	3.50	4.00	4.00	3.40
Grok	compressed	1.50	1.70	1.40	3.20	2.40	3.00
Grok	misleading comments	3.50	3.20	1.00	3.60	2.30	3.00
Sol	clear	4.00	4.00	3.80	4.00	4.00	3.70
Sol	compressed	1.50	1.50	1.50	3.50	3.30	3.50
Sol	misleading comments	4.00	3.80	1.00	3.80	3.40	3.50

All three judges separate the clear program from the compressed one on naming and presentation and penalise misleading comments on intent; the two that are not scored here failed only on the clear control's probe. Full gate values and all ten control means for all three judges are in calibration.json.

Results in detail

Table 4. Code quality components at each effort level

Component	Medium	High	Max
Code quality	65.14	66.26	69.17
Human readability	54.3	57.9	61.4
Maintainability	67.4	67.5	70.8
Intent recovery	76.5	76.4	77.4
Reviewed score	60.9	62.7	66.1
Standard error of Code quality	1.81	2.49	2.91
Judged runs	23	20	22
Runs attempted	23	23	23

Code quality runs 65.14 to 69.17. The rise is carried by the reviewed layer, whose human readability sub-score climbs 7.0 points from medium to max, while intent recovery, the ground-truth layer, stays flat at 76.4 to 77.4. With standard errors of 1.81 to 2.91 the medium to high step sits inside the noise. One judge produced every one of these numbers.

Table 5. Four factors by effort, mean score out of 100

Model	Effort	Functional	Lint, complexity	Security	Code quality
SWE-2	Medium	91.80	79.47	97.39	65.14
SWE-2	High	89.81	79.09	93.25	66.26
SWE-2	Max	97.98	77.83	90.68	69.17

Functional scores retain partial credit, so they sit above the whole-task pass counts. Lint and complexity and security are the sweep's automated measurements. Code quality is the protocol's score. Every mean covers the judged runs in the cell.

The four unjudged runs

Effort	Task	Finished	Reason recorded in the population record
High	cellarcore	no	Reached the 3-hour task budget before verification, so the run has no receipt
High	snapcore	yes	Changed no recognized source file, so there is no submission to judge and the sweep's automated quality and security metrics are undefined
High	vaultcore	yes	Changed no recognized source file, so there is no submission to judge and the sweep's automated quality and security metrics are undefined
Max	freightcore	yes	Changed no recognized source file, so there is no submission to judge and the sweep's automated quality and security metrics are undefined

Every one of the four scored 0 functionally and counts as a fail, so the sweep's pass counts (15, 15 and 21 of 23) are unchanged by the exclusions. The three that finished keep their runtime and tokens in every economics figure; the one that did not has no receipt and its 3.0 hours sit outside the totals. Under v3.11 Muse Spark 1.3 made 204 counted calls: 65 primary reviews, 3 repeats, 6 pairwise checks, 65 intent probes and 65 answer-key matches. No call needed a second attempt, no operator rule was invoked and no reviewer fallback occurred. The earlier v3.9 and v3.10 passes, including the two failed calibration exams, stay archived in the harness run directories.

Cost and tokens across effort levels

Cost per task is unavailable, and that is the published value. Cognition publishes no per-token rate for SWE-2. The cost tier "Free" in Devin's catalog is a promotion dated through 2026-10-10, not a rate, so a figure taken from it would read as a measured price and would stop being true when the promotion ends. No cost is estimated on this page or in the evidence bundle, the population record carries a null cost for every run, and SWE-2 is not compared on cost with the priced models on this board. If Cognition publishes a rate, these runs can be repriced from their receipts.

Devin's own credit and ACU counters in the CLI receipts read zero on every run and are recorded per run, but a zero counter on a subscription is a counter and not a price. What the sweep does spend is time and tokens, and it spends a great deal of both. Neither follows the effort ladder: medium is the heaviest level in tokens, high the lightest and the fastest, and max the slowest. Against the Codex models measured on the same suite, which run about 10 to 12 minutes and 1.8M to 2.8M tokens per task, Devin is several times slower and several times heavier.

Table 6. Tokens, runtime and cost by effort

Effort	Runs	Output/task	Tokens/task	Level tokens	Min/task	$/task
Medium	23	220K	16.80M	386M	49.8	unavailable
High	22	137K	7.53M	166M	35.1	unavailable
Max	23	171K	9.73M	224M	56.6	unavailable
Full sweep	68	177K	11.41M	776M	53.7 h	unavailable

Runs here are the finished runs of each cell, 68 in all; the run that did not finish has no receipt. Tokens are the Devin CLI's own per-request usage receipts, deduplicated by request id and summed as uncached input, cache reads and output. Per-run records, including Devin's credit and ACU counters, are in economics.json and runs.csv.

Limitations recorded with the measurements

• SWE-2 has no public API price, so no cost is estimated and the cost column reads unavailable; the sweep ran on a Devin subscription. The catalog's Free tier is a promotion dated through 2026-10-10, not a published rate.

• Tokens are the Devin CLI's per-request usage receipts, deduplicated by request id: uncached input, cache reads and output.

• Devin's own credit and ACU counters are recorded from the receipts and were zero for every run of this sweep. A zero counter on a subscription is not a price, so it is reported as a counter and not as a cost.

• The 68 finished runs are all counted here, the three that changed no source file included. The one run that did not finish has no receipt and is listed separately; its 3.0 hours of wall clock are not in these totals.

• Runtime is solver wall clock as the harness recorded it, judging excluded.

Evidence and reproduction

Public files: `runs.json` and `runs.csv` with every finished run's factors, the judge's six-dimension sub-scores and intent recovery for each published run, and the four exclusions with their reasons; `groups.json`; `calibration.json` with all three verdicts, every gate value and control mean; `judge-protocols.json` with the exact rubric, system text, probe and match instructions, schemas, weights, the single-panel rule and the population record; `economics.json` with token and runtime aggregates and the record of why cost is unavailable; and `provenance.json` with source hashes and limits.
https://github.com/morganlinton/VulcanBenchCOM/tree/main/assets/data/swe-v4-devin-swe2-v311

Withheld: raw prompts and responses; submitted patches and reconstructed sources; quirk answer keys (they describe hidden-test behaviour); reviewer session identifiers and usage receipts.

Recompute the published numbers

Each run's Code quality is (0.24 x reviewed score + 0.09 x intent recovery) / 0.33, both terms from the single scored judge, or the reviewed score alone where the submission passed no quirk family. The combined score is 100 x (0.50 F + 0.085 Q + 0.085 S + 0.33 C / 100) with F, Q, S on a 0 to 1 scale. Group means weight the cell's judged tasks equally. The export script recomputes every row from the frozen summary, asserts that the four excluded runs are exactly the four the population record lists and that each is a functional fail, asserts that no run carries a cost figure, and refuses to write if any value differs.

What this report does not claim

Code quality here is one model's judgment for a human reader, without the second opinion every other Frontier v4 entry carries and without any human validation. Standard errors describe task sampling only. The measured-maintenance layer is unbuilt, so the 33 points are 24 reviewed plus 9 intent recovery. Runs were not repeated and effort labels are Devin's own. The high cell's Code quality and combined score are means over 20 of 23 runs and the max cell's over 22 of 23. Hash checks bind the export to frozen files; they do not prove that the judge was unbiased or that no training overlap exists.

Source hashes

Frozen artifact	SHA-256
v3.11/summary.json	f3f605603520e6f6941bd605eb1ef79a…
v3.11/protocol.json	a46cda0b67a8480569f0a323a160b86a…
v3.11/private-manifest.json	d9eae1d5ad785c53004b51e9a019c638…
v3.9/calibration-muse.json	390377839156e13d3b66b161f4f06435…
v3.9/calibration-grok.json	0dd725e7226b9ba365fed8311109d002…
v3.10/calibration-sol.json	b6a1b2fb849d1867a5ea69279214ef2f…
comparison-judged.json	e93d351eb4a2257d015c3e329ed6ecac…
comparison.json	9473e196c55443be8daab9cd1cbdd975…

The protocol documents, controls, quirk-key format, runner and operator wrapper are in the VulcanBench repository under docs/judging and harness. The quirk answer keys themselves describe hidden-test behaviour and are not published.

Appendix. Per task at medium and max effort

Task	Medium CQ	Max CQ	Medium combined	Max combined
blendcore	63.8	61.2	86.52	85.72
cellarcore	64.7	40.1	79.09	76.71
codeccore	72.7	77.1	59.32	90.49
depotcore	47.5	57.0	73.27	83.43
freightcore	66.7	excluded	87.30	excluded
granarycore	56.4	57.9	79.21	83.26
hedgecore	63.6	66.7	86.33	87.24
lodgecore	57.6	71.2	75.59	87.48
matchcore-order-book	60.6	77.3	84.95	90.72
metercore	79.4	68.8	91.76	88.29
pacecore	76.5	89.0	85.19	95.06
paddockcore	44.5	40.6	54.55	77.36
payrollcore	59.1	60.3	84.71	77.59
qlite-store	72.7	83.3	89.31	93.02
queuecore	63.6	63.4	73.88	86.06
quotacore	68.2	74.2	88.20	89.62
reflowcore	63.0	56.9	86.44	84.15
schedcore	66.0	67.7	86.89	87.35
settlecore	63.6	87.9	85.95	94.31
snapcore	70.3	74.8	88.83	90.04
stampcore	71.4	81.6	88.97	92.41
tallycore	67.3	79.9	87.95	83.33
vaultcore	78.8	84.8	91.69	71.42

Per-task values from runs.json. Task names are shortened for width. The freightcore run at max is excluded for the reason given on page 4; the two other finished exclusions and the unfinished one are at high effort.

VulcanBench

September 2026

VulcanBench Frontier v4: Devin SWE-2 across effort levels

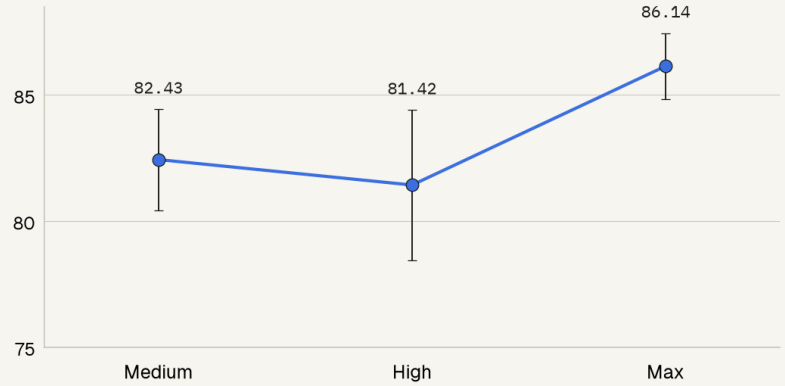
Combined score and runtime at every effort level SWE-2 offers, 23 tasks per effort. Code quality judged by Muse Spark 1.3 alone; see notes.

● **Devin SWE-2 · Devin CLI**

n=23 at medium, n=20 judged at high, n=22 judged at max

Combined score

/100 · higher is better · focused scale



Mean runtime

Minutes per task · lower is better

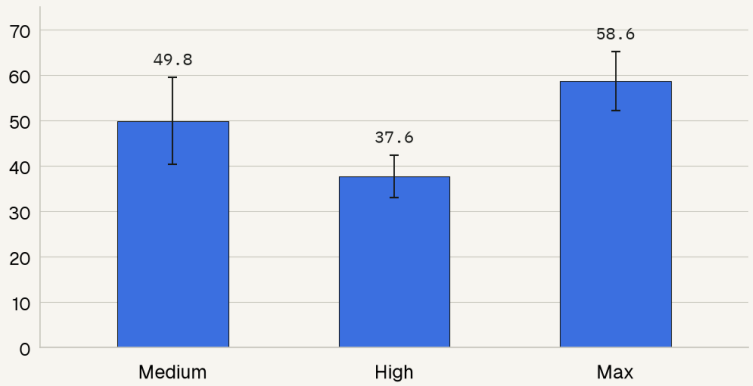


Table 1 | Code quality at each effort level

Component	Medium	High	Max /100, Muse Spark 1.3
Code quality	**65.14**	**66.26**	**69.17**
Human readability	54.3	57.9	61.4
Maintainability	67.4	67.5	70.8
Intent recovery	76.5	76.4	77.4
Rated by Muse Spark 1.3	60.9	62.7	66.1
Standard error of Code quality	1.81	2.49	2.91

Judged on 65 of 69 runs: one high run (cellarcore) hit the 3-hour budget before verification, and three runs (snapcore and vaultcore at high, freightcore at max) changed no source file at all; the protocol excludes those four rather than judging them and the sweep counts every one as a fail. SWE-2 offers medium, high and max only, and nothing is priced: Devin publishes no API rate for SWE-2. Panel differs from the rest of the board: Grok 4.6 (v3.9) and GPT-5.6 Sol (v3.10) both failed calibration gate 16, invented departures on the clear control, so Code quality here is Muse Spark 1.3 alone rather than the two-judge mean behind every other Frontier v4 entry.

VulcanBench September 2026

VulcanBench Frontier v4: Devin SWE-2 across effort levels

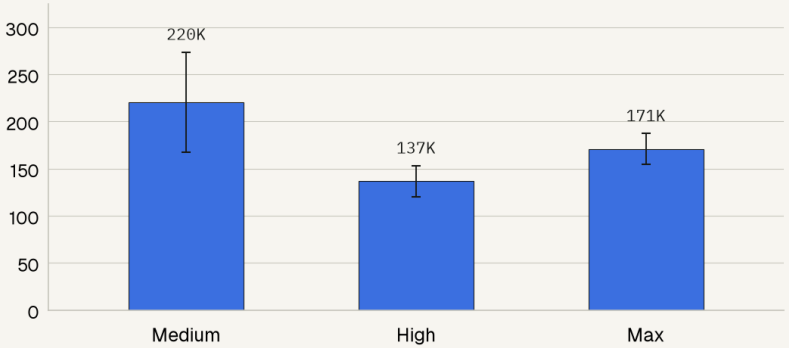
Token use and runtime at every effort level SWE-2 offers, 68 finished runs; solver inference only, judging excluded. SWE-2 has no public API price.

● **Devin SWE-2 · Devin CLI**

n=23 at medium and max, n=22 at high

Completion tokens per task

Thousands of output tokens, the model's own writing · lower is better



Tokens per task

Millions of raw tokens, cache reads included · lower is better

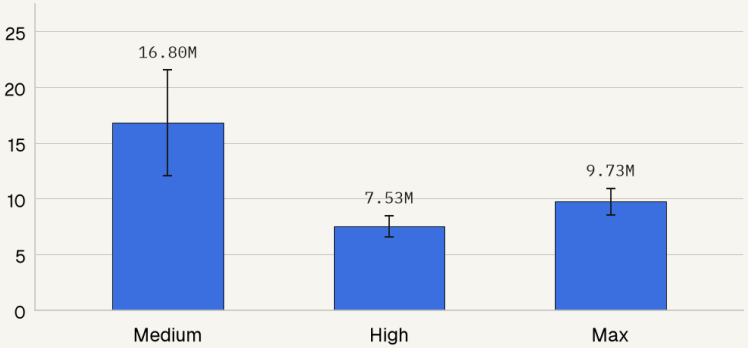


Table 1 | Tokens and runtime at each effort level

Effort	Runs	Output/task thousands	Level total raw tokens, millions	Tokens/task raw, millions	Minutes/task solver wall-clock
Medium	23	220K	386M	16.80M	49.8
High	22	137K	166M	7.53M	35.1
Max	23	171K	224M	9.73M	56.6
Full sweep	**68**	**177K**	**776M**	**11.41M**	**53.7 h**

No cost is shown: Devin publishes no API price for SWE-2 and the sweep ran on a subscription. Devin's credit and ACU counters in the receipts were zero for every run.
68 of 69 runs are here: cellarcore at high reached the 3-hour task budget before verification. Three more runs changed no source file (snapcore, vaultcore, freightcore) and carry no Code quality score, but their tokens and time are real and are counted.
Whiskers are one task standard error. Tokens are raw solver totals including cache reads.